Advanced Machine Learning
Spring, 2018

Deep Kalman Filters

Chitwan Saharia 150050011
Harshith Goka 150050069
Varshith Sreeramdass 150050084
Vishwajeet Singh 150050046

Supervisor:
Prof. Sunita Sarawagi

Saturday 5th May, 2018

Contents

1 Main Goals

The aim of the project is to explore deep sequential models for the task of time series forecast. The main models explored include Deep Kalman Filters and LSTM models with different architectural variations. The dataset used for this project is the Rossman Sales Dataset from this **Kaggle competition**.

2 Relevant Literature

Inspired from Kalman Filters[1], the model attempts to represent the transition models using neural networks to be able to capture non linear transitons.[2] The model uses the idea of using a neural network to estimate a tractable posterior distribution and subsequently maximising a lower bound as in Variational Auto Encoders. [3]

3 Data Preprocessing and Embeddings

Unlike the original paper, the project attempts to model the Rossman Sales Dataset.

The dataset requires a lot of preprocessing before feeding it to the model. The information about the features in the dataset have been briefly described below:

- **Id** - an Id that represents a (Store, Date) duple within the test set
- **Store** - a unique Id for each store
- **Sales** - the turnover for any given day (this is what you are predicting)
- **Customers** - the number of customers on a given day
- **Open** - an indicator for whether the store was open: 0 = closed, 1 = open
- **StateHoliday** - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None
- **SchoolHoliday** - indicates if the (Store, Date) was affected by the closure of public schools
- **StoreType** - differentiates between 4 different store models: a, b, c, d
- **Assortment** - describes an assortment level: a = basic, b = extra, c = extended
- **CompetitionDistance** - distance in meters to the nearest competitor store
- **CompetitionOpenSince[Month/Year]** - gives the approximate year and month of the time the nearest competitor was opened
- **Promo** - indicates whether a store is running a promo on that day
- **Promo2** - Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating
- **Promo2Since[Year/Week]** - describes the year and calendar week when the store started participating in Promo2
- **PromoInterval** - describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store

Figure 1: Features of Rossman Store Sales

Some of the key modifications are as follows:

- For some stores, the sales data for a particular six month interval is missing. Those stores have been treated separately in order to refresh the RNN states during those gaps.

- Some fields like CompetitionOpenSinceYear and Promo2SinceWeek are discreet but with very wide range of values, but considering every value makes the distribution sparse. Hence, we performed required binning to reduce the range considerably.
- Only continuous feature is the CompetitionDistance which has been normalised.
- Further details of the dataset preprocessing can be found in data_process.py

4 Approaches Tried

Due to the sequential nature of data, recurrent neural networks formed a major segment of the models implemented. LSTMs and their variations are described in the first section and Deep Kalman Filters are described in the second.

4.1 LSTMs

Basic RNN, though they are able to capture dependence across time steps, can't handle long term dependency. To circumvent the problem LSTM cells are used instead.[4]

4.1.1 Baseline

To rely less on the hidden state and it representing the data robustly enough, the output of the RNN in a particular step is fed to the RNN in the following step as input (in the training phase). This provides more pathways for the training of the parameters.

4.1.2 Teacher Forcing

As opposed to the above method, the actual values of the outputs are fed to the RNN in the training phase in place of the predicted outputs.[5]

4.1.3 Scheduled Sampling

Combining the above methods, the input is a random sample from the actual outputs and the predicted outputs.[6]

4.2 Deep Kalman Filters

Deep Kalman Filters inspired from Variational Autoencoders, attempt to obtain a multivariate gaussian distribution over hidden states of the data at a particular time. The progress of a sampled hidden state over time, and

the generation of data from the hidden state are modelled using feedforward networks.

The recognition model consists of a recurrent network used to train the model and to predict the distribution of data at a particular time supervised by the outputs.

4.2.1 Generation

From a particular hidden distribution at time t obtained using the recognition model, a hidden state z_t is sampled. The generation model predicts a multivariate gaussian distribution from which a sample serves as the output x_t. The hidden state z_t is used to predict the distribution over the hidden state at the next time instance z_{t+1}.

5 Experiments

The following table lists the experiments conducted:

Model	Training Error	Validation Error	Test Error (RMSPE)
Basic LSTM	0.12	0.19	0.24
LSTM Teacher Forcing	0.08	0.1	0.29
LSTM Scheduled Sampling	0.07	0.11	0.27
Deep Kalman Filters	In Progress	In Progress	In Progress

You can find the code here at : https://github.com/chitwansaharia/AMLProject

6 Effort

Chitwan Saharia - Worked on Baseline LSTM Models and Data-Preprocessing
Harshith Goka - Worked on Deep Kalman Filter
Varshith Sreeramdas - Worked on Deep Kalman Filter
Vishwajeet Singh - Worked on Baseline LSTM Models and Data-Preprocessing

References

[1] R. Kalman, "A new approach to linear filtering and prediction problems," vol. 82, pp. 35–45, 01 1960.

[2] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep kalman filters," *arXiv preprint arXiv:1511.05121*, 2015.

[3] D. P Kingma and M. Welling, "Auto-encoding variational bayes," 12 2014.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," vol. 1, 09 1998.

[6] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.